SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NRL Memorandum Report 6026	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Research Laboratory — 6b. OFFICE SYMBOL (If applicable) Code 5720	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Washington, DC 20375-5000	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Office of Naval Research — 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
PE62111N	PE11 E60		EX155-640

11. TITLE (Include Security Classification)

Evaluation of the Very High Speed Microprocessor Breadboard

12. PERSONAL AUTHOR(S)

Christiansen, Ross M.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Interim	FROM ______ TO ______	1987 July 21	30

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Computers, Processors, Microprocessors, EW, ESM, Benchmarks, Gibson Mix, AYK-14

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report covers evaluation of the Very High Speed Microprocessor (VHSM) breadboard performance and application cost effectiveness. Execution rates of 1.7 MIPS for the Gibson mix and 2.9 to 3.6 MIPS for EW applications have been measured. Comparisons are made to various military and commercial computer products. Use of all levels of language programmability, including extended instructions, are demonstrated.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Ross M. Christiansen	22b. TELEPHONE (Include Area Code) (202) 767-2653 — 22c. OFFICE SYMBOL Code 5720

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

Naval Research Laboratory

Washington, DC 20375-5000

NRL Memorandum Report 6026

Evaluation of the Very High Speed Microprocessor Breadboard

R.M. CHRISTIANSEN

Electronic Warfare Support Measures Branch
Tactical Electronic Warfare Division

July ~~21~~, 1987

Approved for public release; distribution unlimited.

CONTENTS

EVALUATION OF THE VERY HIGH SPEED MICROPROCESSOR BREADBOARD

INTRODUCTION

The Very High Speed Microprocessor (VHSM) program has been addressing cost-effective signal processing in electronic warfare (EW) systems.(Ref. 1) Experience over the useful life cycle of numerous computer/processor designs has shown that the cost of software development and maintenance will be many times more than the hardware costs. Studies have shown that software typically accounts for 80% of a computer system's life cycle cost, and this number is expected to grow to 90% by 1990. (Ref. 2) Cost effectiveness in the VHSM has been addressed through the establishment of a microprocessor architecture that achieves optimum throughput efficiency in EW and other applications, a macroarchitecture compatible with existing support software, and programmability by common familiar and military standard higher order languages.

Significant cost savings can be achieved by basing new processor designs on familiar characteristics rather than creating new unfamiliar characteristics. These characteristics are reflected primarily in the instructions executed by the processor, i.e., the macroarchitecture. The VHSM was designed to operate with an instruction set based on an extension of the AN/AYK-14 instruction set. This permits use of Navy standard support software hosted on all popular computer systems and programmability by both popular (FORTRAN) and military standard (CMS-2) higher order langauge. It can be expected that the Navy will eventually modify its standard support software to permit programming of the AN/AYK-14 by Ada. Since the VHSM is compatible with the AN/AYK-14 support software, it too will be programmable by Ada.

Evaluation testing reported here was performed on the first-generation feasibility breadboard (VHSM-1) which was implemented using commonly available discrete devices and a bit slice ALU. Performance was optimized through the incorporation of several hardware speed-up techniques such as instruction pipelining, split memory, and path length minimization. Special consideration during the development of the architecture concept resulted in a technologically tolerant system architecture. A second-generation feasibility breadboard (VHSM-2) is currently being implemented using advanced large scale CMOS gate array technology. (Ref. 3)

Test programs were developed to demonstrate and evaluate numerous features of the VHSM architecture, as well as its basic execution rate performance. VHSM execution performance was evaluated by measuring the execution of a number of test programs on the feasibility breadboard. Comparison has been made to the AYK-14 using the MTASS AYK-14 timing simulation and execution of several test programs on an actual AYK-14 computer. Comparison has also been made to several newer state-of-the-art microprocessor systems.

Manuscript approved April 9, 1987.

The concept of "testing a computer" is quite ambiguous and the results often misleading when taken out of context. Testing procedures are controversial and there are no commonly recognized standard tests which fully define performance. The very concept of "performance" is vague. Most often, performance is simplistically defined as "throughput". Throughput is actually the rate at which transactions are presented to, and disposed of, by a system. This actually involves the broad concept of performing a work function. In many cases throughput is translated into execution rate, which is a more narrow concept.

Computers are used to perform work. Measuring work involves much more than just measuring how fast the computer can execute instructions. In reality, many other features of a computer system determine how much work can be performed. These features have to do largely with the means of performing input and output functions. These are also very complex functions, and often wind up being the real limiting factors in how much work can be performed by a system. The VHSM-1 breadboard was not intended to represent a complete computer system, and was not capable of performing work in this context as it is presently configured. The subject of input/output functions are being addressed in a subsequent phase of the program.

One of the commonest oversights in computer testing is to attempt to measure performance without due consideration of software. Several of these methods have been lumped together as "bad" approaches to first cut performance analysis. (Ref. 4) Two of these "softwareless" approaches involve the use of standard test programs which are commonly used to make side-by-side comparisons between hardware systems. This approach involves the use of instruction mixes and benchmarks. Both of these test formats were used as part of the evaluation program for the reason that they do offer a simple method of comparison between systems.

Besides this use of standard programs, very little source material has been developed in the area of test algorithms to evaluate systems, especially in specific application fields. Several programs considered to be representative of applications were included in the evaluation. In addition, several specific application programs have been obtained from operational systems. To partially compensate for the lack of software impact in the standard comparison programs, several test programs were written in more than one language format to determine software dependence. These samples are too small to be considered representative or to draw conclusions from, but are interesting examples useful in demonstration of the phenomena. Also, one application program was exercised with different data sets to demonstrate the phenomena of data dependence. Again, this sample is too small to be considered representative, but is offered as interesting example.

TEST PROGRAMS

Standard test programs come in two basic forms: instruction mixes and benchmarks. For instruction mixes, the instruction repertoire is divided into several broad classes of instructions, such as: load/store, arithmetic/logical, fixed/floating point, multiply/divide, branch, etc. Each group of instructions is given a weight representative of its occurrence of use in programs. Different mixes are given for different classes of applications. Presumably, if you have an application that is characterized by a particular mix, then a comparison of the instruction mix execution times for two computers gives a fairly direct measure of relative performance in that application. One of the basic problems with instruction mixes is that they are usually derived from static sources, i.e. program listings. Mixes derived from dynamic sources representing actual executed use in programs are obviously more germane, but much more difficult to derive. Two instruction mixes were used in the evaluation program, one representative of general purpose applications and one very application specific.

Benchmarks carry this execution model concept further into more exhaustive programs. A benchmark is a higher level program consisting of a number of lower order programs that, as a set, characterize the processing load. A benchmark is run on the target computer and the performance measured. This can be a useful approach for large scale data processing systems, but is of little value to small systems with unique applications. Popular examples of such benchmarks are the Whetstone and Livermore loops. These benchmarks emphasize floating point operations which were not implemented in the VHSM. (The VHSM-1 hardware performs only fixed point operations.) Two benchmark programs were used in the evaluation, primarily for comparative purposes in a compute-bound environment.

The new fields of digital signal processing have led to new special categories of figures of merit (FOM). These FOMs are derived from functions implemented to perform data flow and transformation computations. These types of operations are widely used and cut across many application fields. Data flow computations are based on a flow of data through processing stages without storage and sequencing constraints. (Ref. 5) Processing is limited to fundamental operations such as addition, subtraction, multiplication, division, and square root. The FOM is complex operations per second (COPS), typically 4 multiplies and 6 adds. Transformation computations arise out of the necessity to use the frequency domain to process large quantities of data and operations (for performing spectral analysis) or to perform filtering. The FOM is seconds per fast Fourier transform (FFT), where the number of points of the FFT can be as large as 1024. These are not totally realistic benchmarks because they ignore the loading of operands and the fetching and decoding of instructions. The above FOMs are intended for evaluating vector processing architectures, not architectures based on programmable ALUs such as implemented in the VHSM.

Outside of the above standard test program category, very little exists in the way of available software routines for use in evaluating computer performance. Several new test routines were created for use in the current evaluation. These programs are algorithms representative of applications encountered in EW and radar systems. These algorithms are generic in nature, and are intended to demonstrate potential usefulness of the VHSM in system applications. In addition, one specific algorithm from an operational system has been converted to run on a machine based on a different architecture concept. In this case, the operation of the VHSM can be compared directly to an operating system.

In all, eight basic programs were developed, encoded in various languages, exercised on instruction simulations, executed on the VHSM breadboard, and executed on a comparison military computer. These programs may be classified as follows:

Mixes

Gibson Mix
EW Instruction Mix

Benchmarks

Puzzle
Statistics

Applications

EW Tracker Algorithm
EW Sort Algorithm
Track Update Algorithm
Radar Data Processing

Gibson Mix

The best-known published example of a standard instruction mix is the Gibson mix. An instruction mix model depends on the architecture of the CPU. The same processing needs may result in different mixes when expressed in languages of different machines. The Gibson mix reduces this dependence by choosing work-load parameters representing logical rather than physical resources. Gibson originally obtained the frequencies in this mix from a large amount of trace data collected in IBM 7090 installations, and it reflects usage in scientific and technical applications. Table 1 presents the original Gibson mix, which is a set of weights developed for 13 different classes of instructions. (Ref. 6)

Table 1

Original Gibson Mix

Instruction Class	Percent
Load and Store	31.2
Fixed Point Add/Sub	6.1
Compare	3.8
Branch	16.6
Floating Add/Sub	6.9
Floating Multiply	3.8
Floating Divide	1.5
Fixed Point Multiply	0.6
Fixed Point Divide	0.2
Shifting	4.4
Logical	1.6
Non Register Instruction	5.3
Indexing	18.0
	100.0

Quite often the Gibson mix is presented in a condensed version of 8 classes of instructions. This simplification is achieved by merging the following classes: fixed point (add/subtract/multiply/divide), floating point (add/subtract/multiply/divide), and shift/logical. (Ref. 7)

The Navy has long used a modified fixed point version of the Gibson mix in evaluating and rating its standard computers. The modified Gibson mix is derived by applying the weights of the three floating point classes of instructions to the equivalent fixed point classes of instructions, as indicated in Table 2. Since the VHSM-1 breadboard did not implement floating point instructions, this modified fixed point version could be used directly as a test program. Since an instruction mix is nonexecutable, it was necessary to convert the weighted instruction set into an executable program format. The final executable program consisted of a sequence of instructions and program loops that, when executed, resulted in 100 instructions being executed with the weights indicated in the table.

Table 2

Fixed Point Gibson Mix

Instruction Class	Percent
Load/Store	31
Add/Sub	13
Compare	4
Branch	17
Multiply	4
Divide	2
Shift	4
Logical	2
Non Reg. Instr.	5
Indexing	18
	100

EW Instruction Mix

To provide a baseline for the original VHSM architecture design and an initial projected performance, a study was performed to establish a representative EW instruction mix. The method used was the determination of instruction weight factors extracted from a range of EW application programs. It was of particular interest to determine if different types of functions within the EW system might significantly impact the instruction mix. The resultant difference in operations between system functional levels was not significant in the systems studied. Table 3 presents the results of this software study. (Ref. 8)

Table 3

EW/ESM Instruction Mix

Instruction	Weight
Arith add/sub (RR)	0.17
Logical (RR)	0.16
Cond Branch (D)	0.14
Load (DX)	0.12
Store (DX)	0.10
Quick Branch (D)	0.06
Mem to Reg (DX)	0.05
Load (D)	0.04
Store (D)	0.04
Branch to Sub (D)	0.04
Double Prec (RR)	0.04
I/O	0.02
Others	0.02

During the study, special effort was made to distinguish between direct (D) and indexed (DX) instructions. This distinction has not been traditionally made in previous studies of this nature, despite the fact that execution time is significantly dependent on the modes of instruction addressing. Table 3 shows that register-to-register (RR) arithmetic and logical instructions make up fully one-third of the instruction mix. The table also shows that the 10 most used instruction classes account for 92%, and the 12 most used instruction classes account for 98% respectively, of the instructions used in the EW applications studied.

Again, it was necessary to convert the above weighted instruction mix into an executable program format. Since the VHSM-1 breadboard does not support the input/output functions, it was necessary to slightly modify the mix of instructions to create an executable artificial program model representative of the EW weighted instruction mix. Table 4 is the actual instruction list incorporated in the executable model.

Table 4

EW Benchmark Instructions

Instruction	Number
Arith add/sub (RR)	17
Logical (RR)	16
Cond Branch (D)(not taken)	7
Cond Branch (D)(taken)	7
Load (DX)	12
Store (DX)	13
Quick Branch (D)	7
Mem to Reg (DX)	5
Load (D)	4
Store (D)	4
Branch to Sub (D)	4
Double Prec (RR)	4
Multiply (RR)	1
total	101

Puzzle

Puzzle is one of a set of four programs referred to as the Berkeley Benchmarks. They were originally put together by Professor David Patterson of U.C. Berkeley and used as the basis of several articles published in Computer Architecture News in 1982. Puzzle is "an undocumented compute-bound program from Forest Baskett." It was chosen for use in the evaluation because it measures the execution speed of a CPU on integer arithmetic, indexed operations, procedure calls, and looping. The orignial source listing was provided in Pascal, and rewritten here in AYK-14 assembly language . (Ref. 9)

Statistics

Statistics is a new program extracted from an application subroutine used to compute the means and standard deviations from a library of parameter measurement data. It is considered to be representative of a practical application requiring extensive statistical analysis of engineering data. It was chosen for use in the evaluation because of the representative nature of the application, and because it had been originally developed and was in use on an operational AYK-14 computer. In the version used here, the program provides for analysis of an array of 1024 measurement samples. No specific data table was created, since the algorithm consists of fixed non data dependent operations. The program was written in AYK-14 assembly language.

EW Tracker Algorithm

The EW tracker algorithm is typical of an embedded processor application for tracking emitter pulse trains. The algorithm technique is sometimes referred to as a table driven tracker because the predicted parameters for signals under track are organized in data tables or arrays. Use of table organized data was cited as one of the gross architectural features representative of EW system implementations that influenced the architecture design of the VHSM. (Ref. 1) For purposes of the evaluation, the algorithm was implemented with a 25 point predict table and a 16 point data table. Four different data tables were created to represent four unique possible conditions of track data. The algorithm is very decision oriented, another EW representative feature influencing the VHSM design. Because of this feature, this algorithm serves as a useful example of data dependent variations in execution rate performance. The algorithm was encoded in two different language formats (AYK-14 assembly language and CMS-2) to also permit comparison of language dependent variations.

EW Sort Algorithm

The EW sort algorithm is typical of an embedded processor application for separating interleaved emitter pulse trains into different bins. Comparison to signature library files is made based on a three parameter signature window. Signatures that successfully match the library values undergo additional processing to predict the next expected time of arrival. The algorithm was encoded in two different language formats. Besides the basic version coded in AYK-14 assembly language, a second version was coded using several of the supplemental (EW) instructions developed just for the VHSM.

Track Update Algorithm

The track update algorithm is a different form of track algorithm typical of an application in a hybrid EW signal processor. The hybrid processor is a potentially higher throughput design employing dedicated hardware at the input stages to match or block input pulse descriptor words. The hardware may measure/quantify parameter error and pass such information back to the processor for update of the signature parameters for signals under track. The algorithm was adapted from an actual microprogrammed track processor. For purposes of the evaluation, the algorithm was implemented to receive error measurements on a three parameter signature, update the three parameter signature files, and predict the next time-of-arrival. A 10 point input data table was implemented. Three different data tables were created to represent three unique possible conditions of track data, to again demonstrate data dependent variations. The algorithm was written using several of the supplemental instructions for the VHSM, and in assembly language for the AYK-14.

Radar Data Processing

The radar data processing algorithm is representative of an embedded processor application to perform post detection processing in radar systems. Threshold crossing data from a non-coherent detector is processed to perform an M out of N detection, centroid the resultant detections, and format the centroided detections for hand off to a tracking system. The algorithm was adapted from an actual experimental radar system.

EXECUTION RATE PERFORMANCE

Observability is a fundamental problem in testing computers. Every act of measurement introduces artifact, which is the perturbation to a measurement brought about by the act of measurement. In evaluating the VHSM, it was desirable to measure the rate of execution of instructions without slowing down the actual execution process. Normal computer operation is accessed via an I/O channel. If measurement of execution is processed via an I/O channel, the I/O operation becomes part of the measurement. If I/O operation is not part of the test, it then represents an error in the measurement.

Hardware monitors cause a minimal amount of artifact, but are limited in the complexity of measurement that can be detected. For the purposes of measuring software execution, it is sufficient to simply detect the occurrence of an appropriate program instruction. A logic analyzer was connected to the VHSM program counter, and programmed to trigger on an appropriate (qualifier) instruction. Most logic analyzers have the ability to generate an external strobe pulse when triggered. The strobe pulse in turn can be used for traditional measurement of the time interval.

In most cases, execution performance has been expressed as a rate of execution of instructions. It is derived by measuring the time required to execute a specific instruction loop of known length, and computing the average effective rate of execution. That rate is expressed as millions of instructions per second, or MIPS. Figure 1 shows the measured execution rate for the two instruction mixes and two application algorithms. The benchmark programs (Puzzle and Statistics) are compute bound programs involving too many instruction cycles to tally. Their measured execution times are presented in Figure 2.

Execution rate was not computed for the two remaining test programs. Execution of the track update algorithm involves significant data dependence, which makes estimation of the actual instruction count difficult. The radar data processing program was written in FORTRAN and demonstrates the use of a popular higher order language in an embedded processor application. Since address locations cannot be specified in FORTRAN, memory configuration of the VHSM could not be optimized. The execution time for the 150 point data table was 1995 usec.

The VHSM architecture includes a configuration feature which can affect performance. That feature has to do with the ability to allocate program and data in the split memory. Program execution rate will differ depending on whether the data to be accessed is in the same or alternate section of memory. Except as noted, the above performance data represents the faster split memory configuration.

Figure 3 shows a comparison of execution performance for the two memory configurations. The amount of improvement that can be achieved depends very much on the type of application. A 7% improvement was achieved with the Gibson Mix, and 11% improvement was achieved with the track update algorithm. However, a similiar comparison with the Statistics benchmark (not shown) showed only a 1% improvement.

COMPARISON TO THE AYK-14

Operation of the AYK-14 is subject to the same observability problem noted above. Because the AYK-14 was operational equipment, it was decided not to disassemble and instrument the actual hardware. Instead, appropriate breakpoints were set in the test programs to read the computer clock register. Due to an initial hardware availability problem, execution of some routines was also evaluated using a non-hardware AYK-14 configuration. This technique involved use of the AYK-14 timing simulation model which is part of the MTASS package.

Figure 1: VHSM Execution Rate
(Split Memory Configuration)

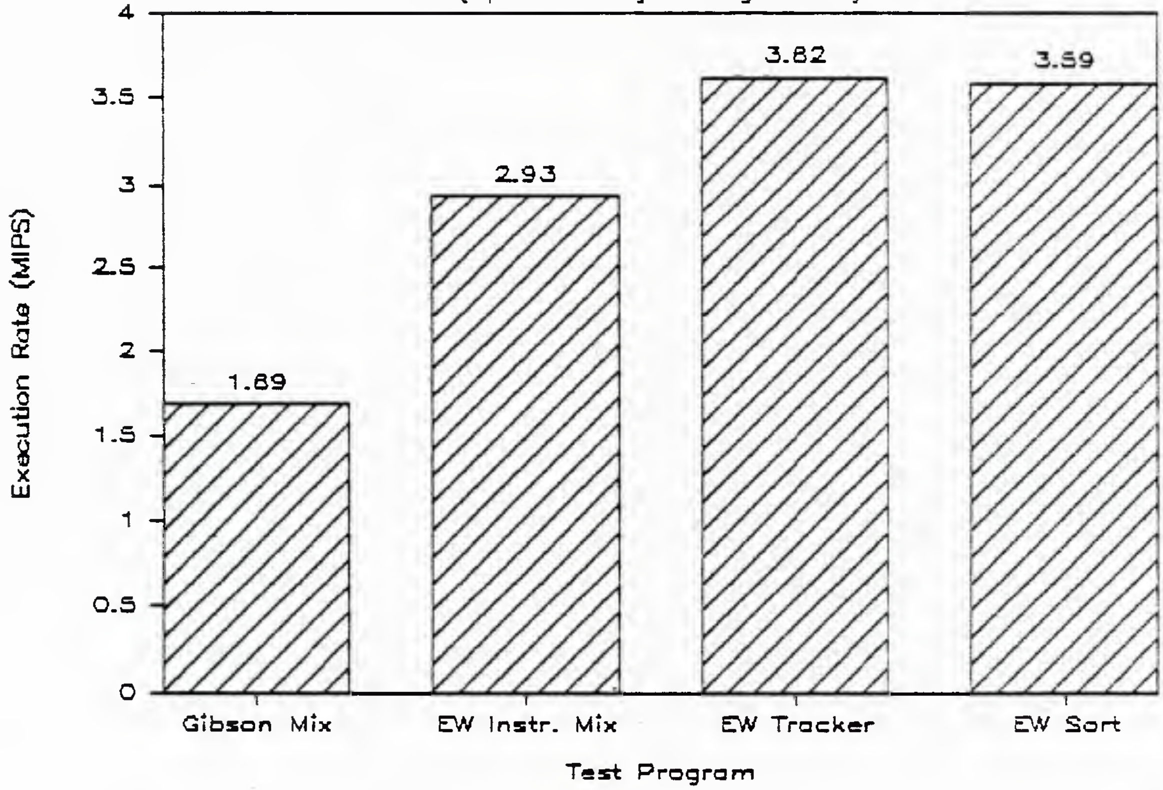


Figure 2: VHSM Benchmark Execution
(Split Memory Configuration)

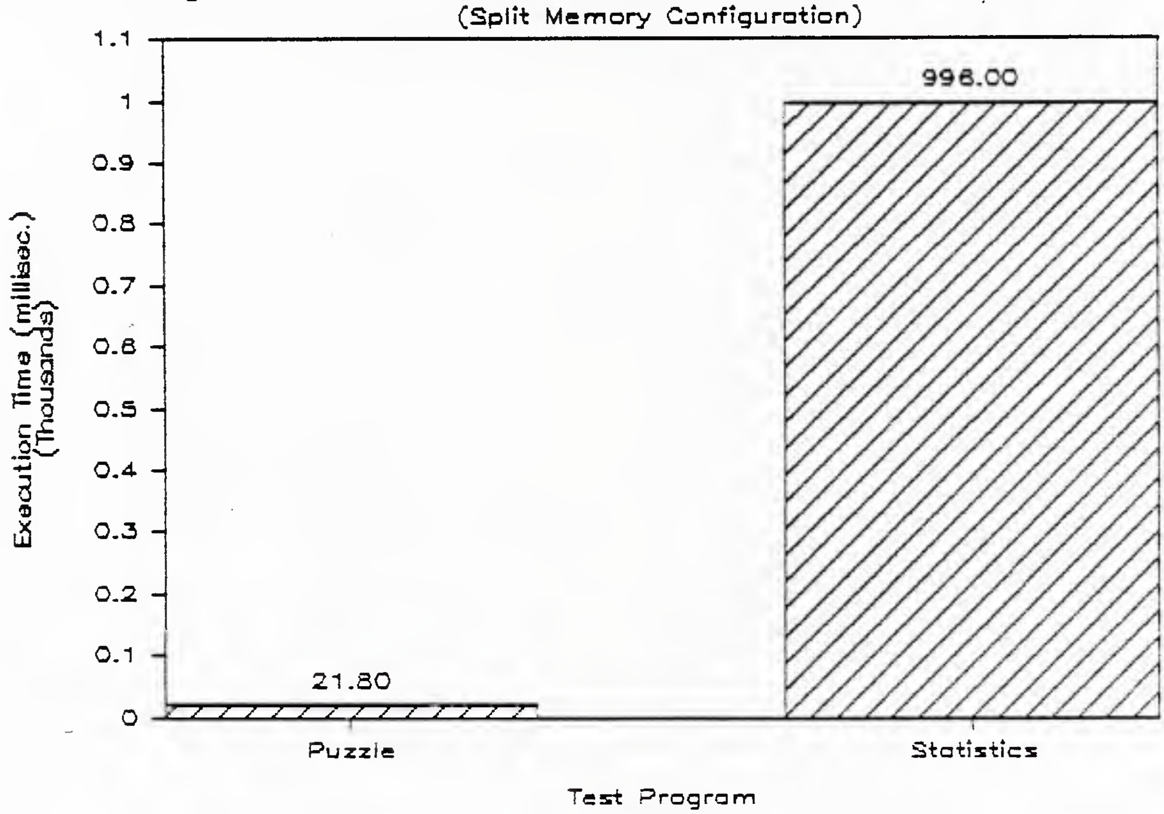


Figure 3: Memory Allocation Comparison

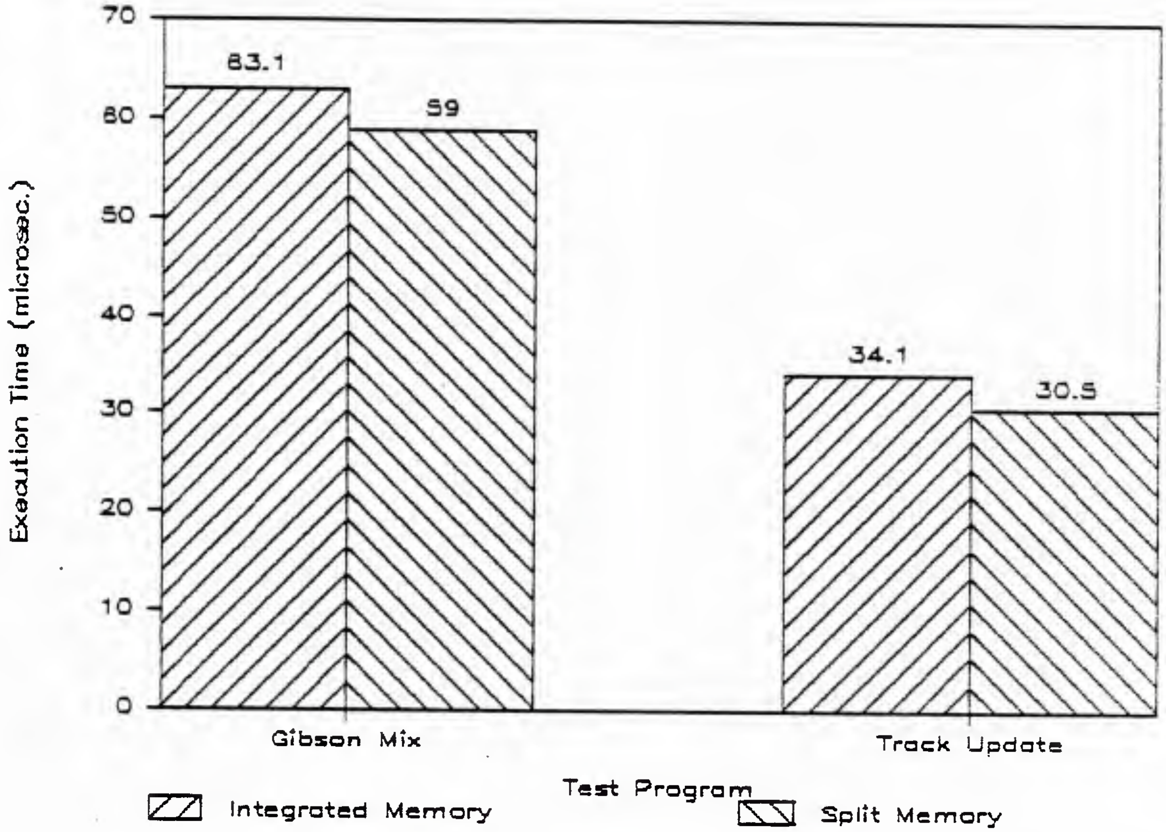


Figure 4: AYK–14 Execution Comparison

(Hardware Execution Rate – Gibson Mix)

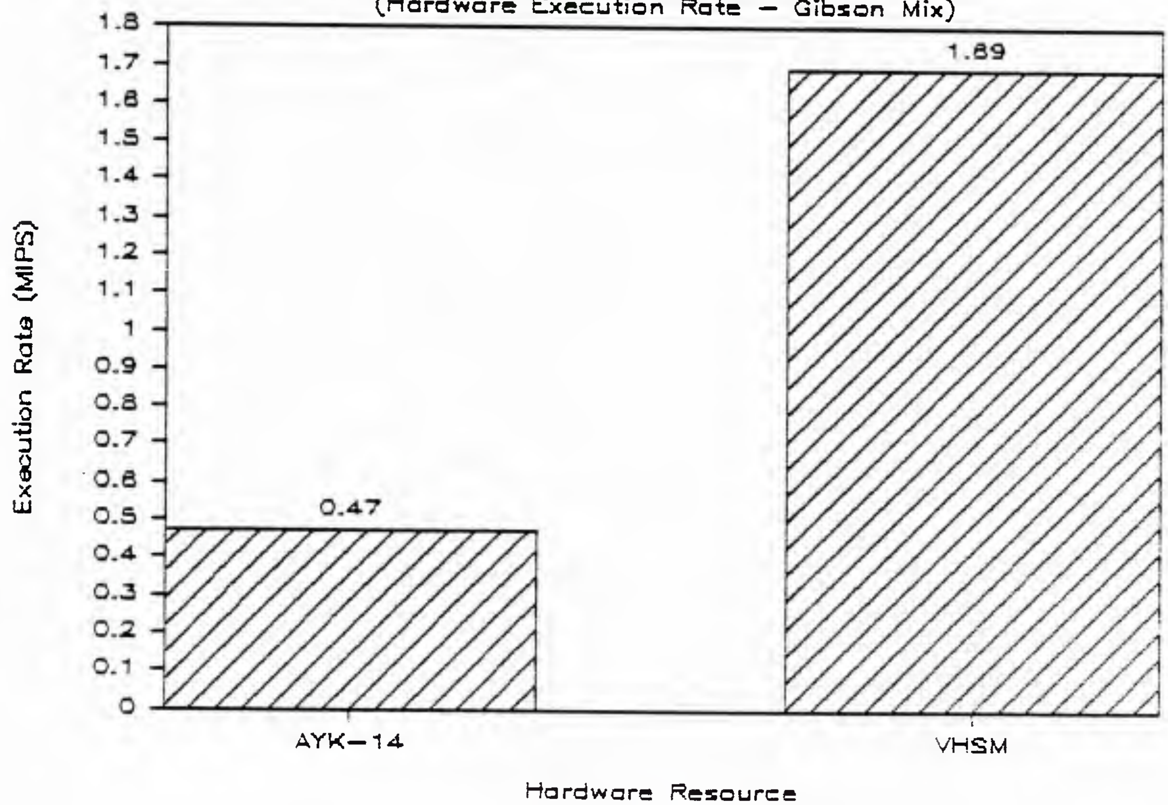


Figure 4 shows comparative hardware execution rates for the Gibson Mix. Figure 5 shows comparative hardware execution times for the Puzzle benchmark and radar data processing. The improvement factor in hardware execution ranged from 3.6 to 4.9. A similiar comparison based on execution of the Statistics benchmark (not shown) showed only a 3.0:1 improvement. Figure 6 shows comparative hardware execution times for the track update algorithm. Slightly different versions of the algorithm were exercised in this comparison. Several AYK-14 assembly instructions were replaced by new instructions in the VHSM version. The improvement factor in this comparison was 4.6

Figure 7 shows the execution rate comparison resulting from use of the AYK-14 MTASS simulation. The EW tracker algorithm was evaluated in both assembly language (Tracker/AL) and higher order language (Tracker/HOL) formats. Execution improvement factors in this comparison ranged from 4.3 to 5.4.

COMPARISON TO OTHER PROCESSORS

Development of the VHSM was initiated in 1980 to fill a perceived void for a high performance embedded processor. Processor development has continued to receive much attention and investment both in the military and commercial market. The initial architecture studies projected performance against then identifyable or projected military processors. Since then, new military programs have been initiated leading to new standard military processors. The commercial market has mushroomed and produced several major series of very advanced processors. These products are also being accepted as embedded processors in military systems.

A natural question arises as to whether or not the original performance projections against older products is sufficient to justify use of the VHSM. Commercial processor evaluation is controversial and the test programs usually cited differ significantly from the programs used in the VHSM evaluation . The following comparisons to newer products has been prepared using the best available information. Comparisons are made to the AYK-14 Single Card Processor (SCP) and the National Series 32000 CPUs. For completeness, a comparison is also made to the standard for high performance processing, the bi-polar bit-slice microprocessor, and the first programmable VHSIC signal processor.

AYK-14 Single Card Processor (SCP)

The SCP is a newer, enhanced instruction version of the AYK-14 architecture. It is an improved performance processor resultant from the AYK-14 Pre-Planned Product Improvement (PPPI) program. The implementation technology is state-of-the-art CMOS gate array, very similar to the technology being used to fabricate the second generation VHSM. Availability is currently very limited, preventing any actual hardware evaluation.

Figure 5: AYK-14 Execution Comparison

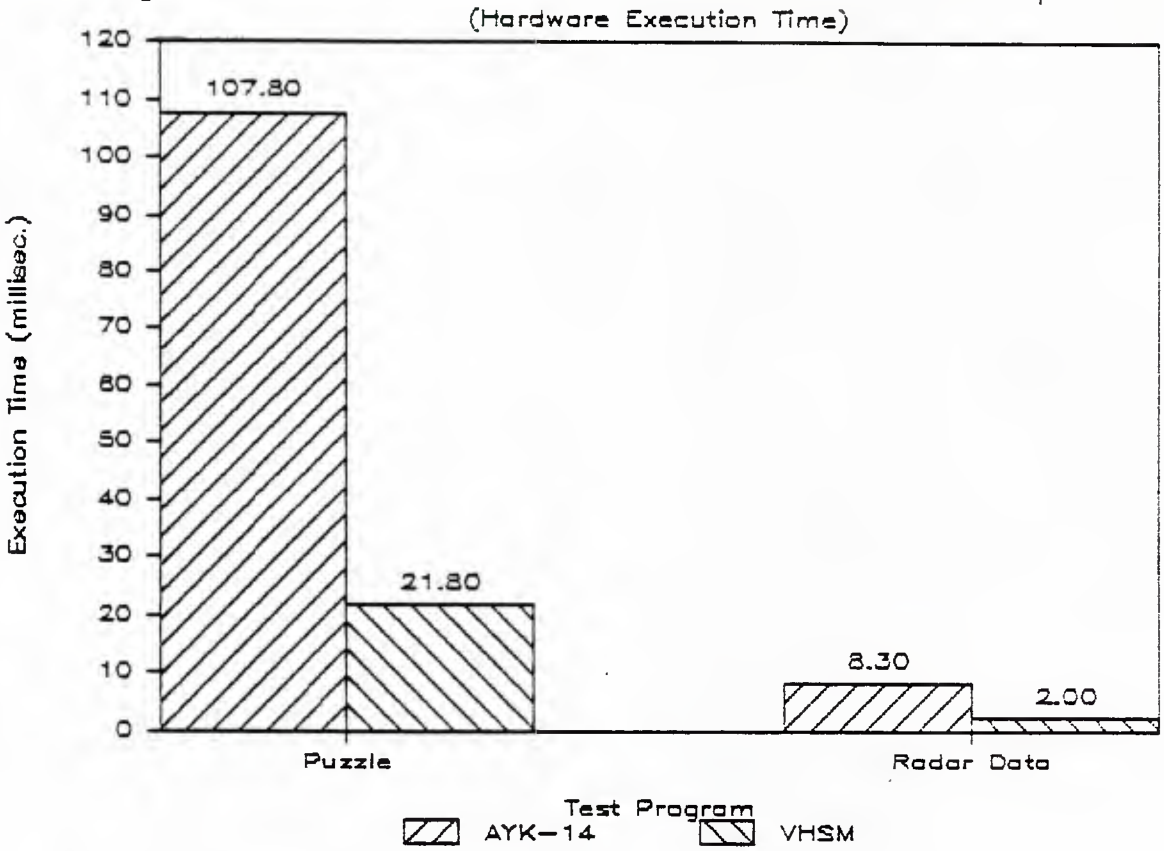


Figure 6: AYK-14 Execution Comparison

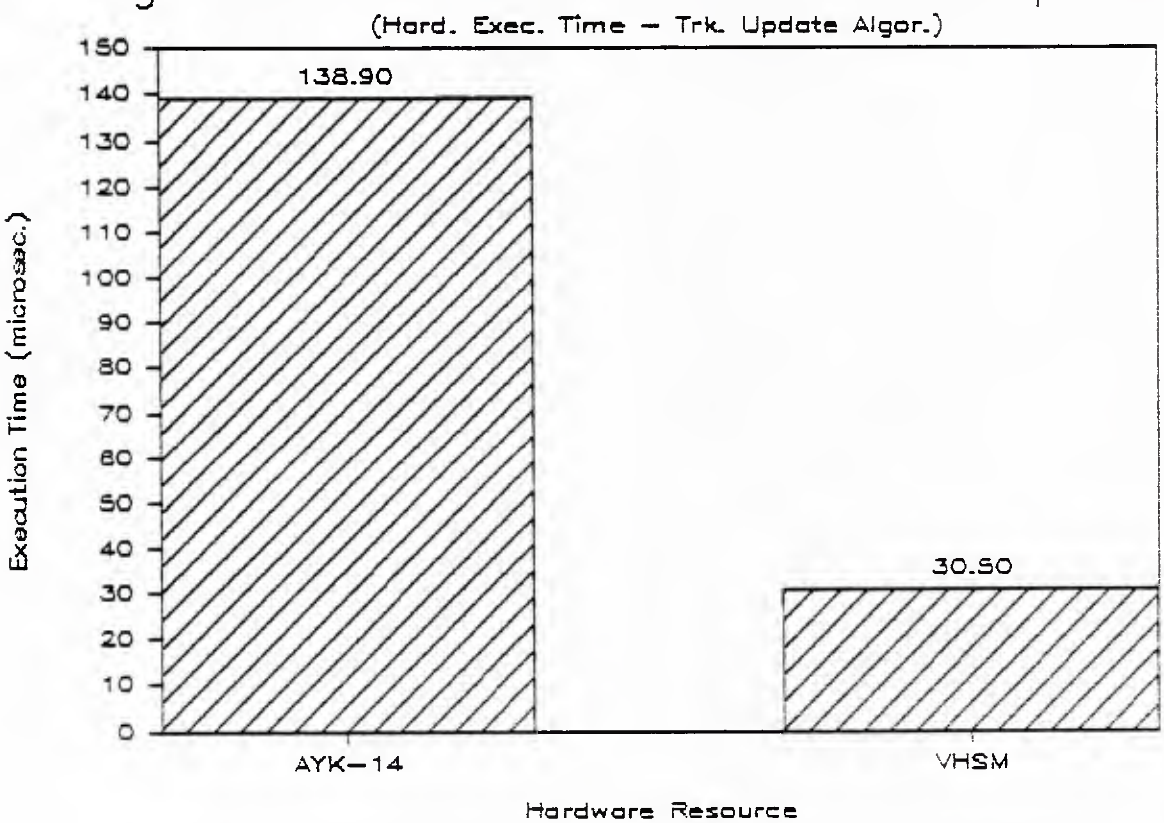


Figure 7: AYK-14 Execution Comparison

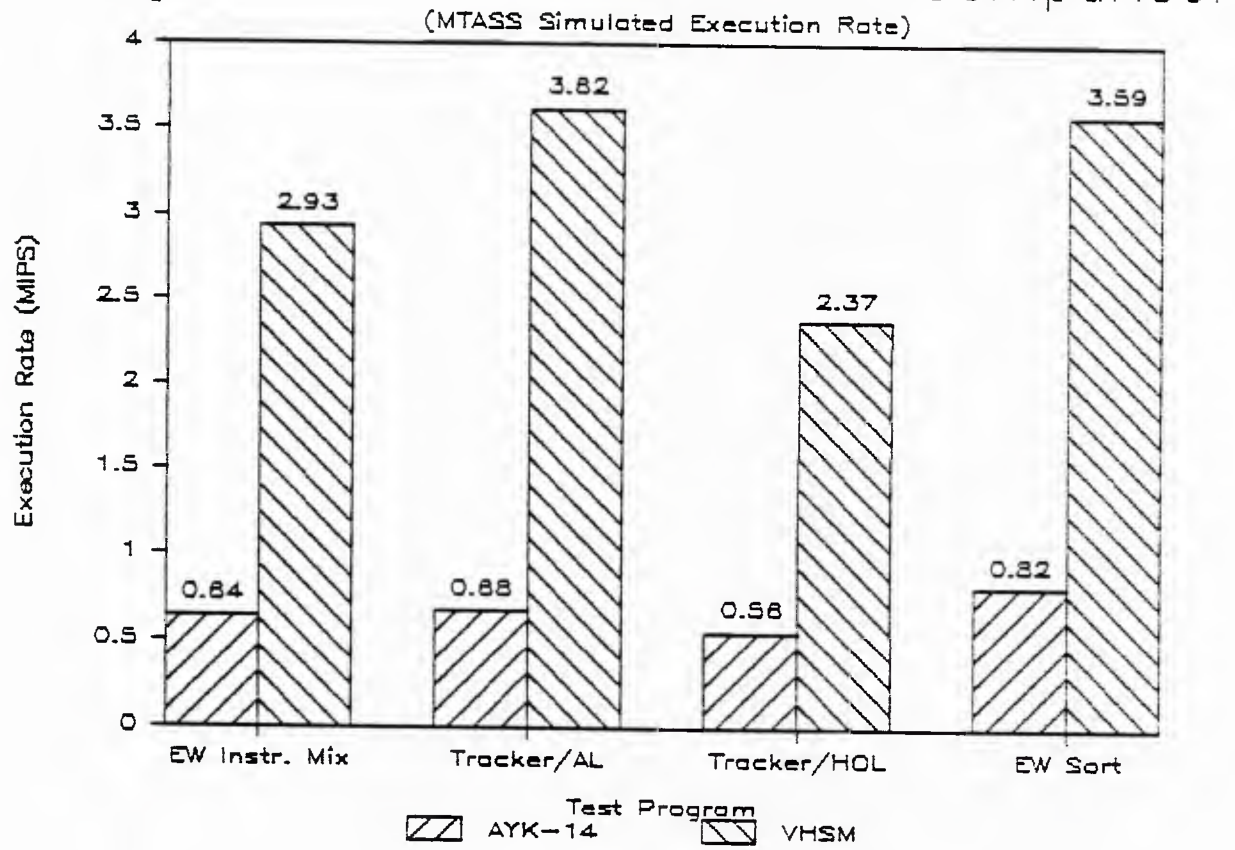
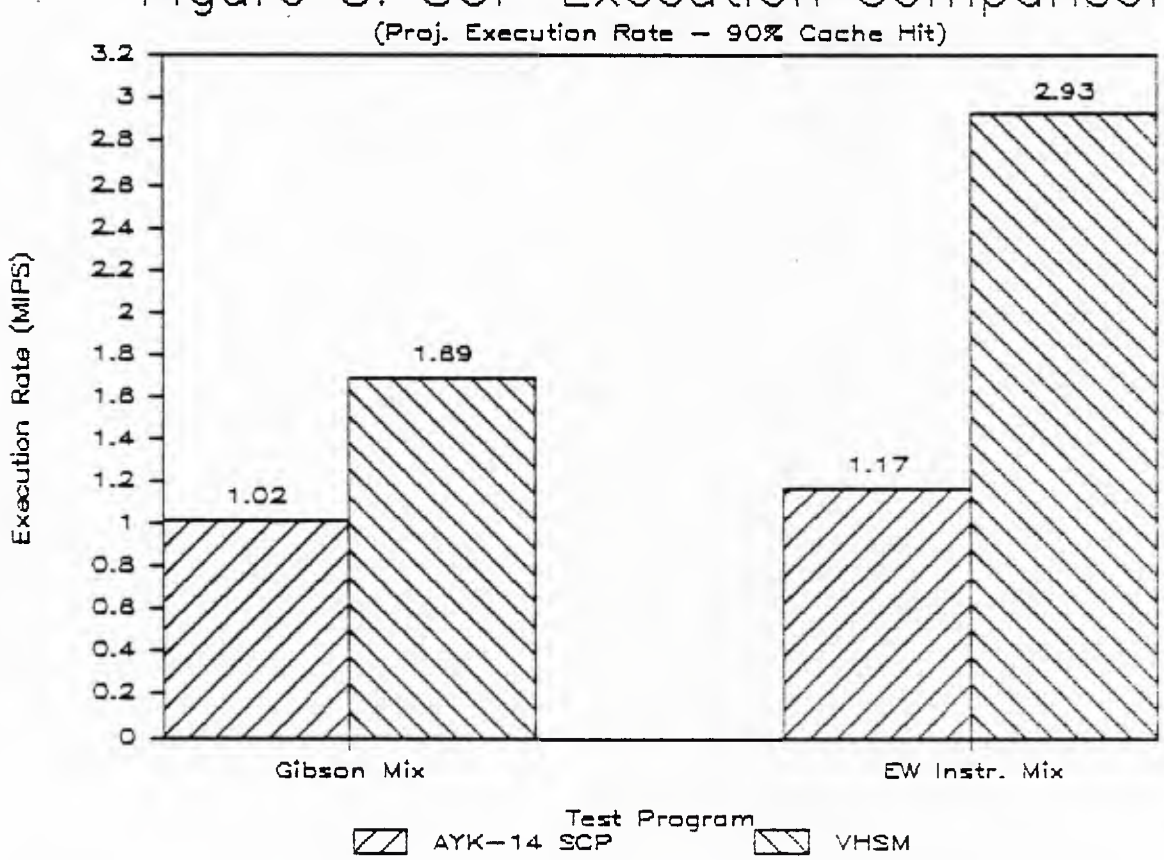


Figure 8: SCP Execution Comparison



Performance projections have been prepared using execution times available in the public domain. (Ref. 11) The SCP architecture includes cache memory for very fast memory access time. The execution times are considered preliminary and are based on an assumed 90% cache hit rate. Figure 8 compares the execution rate for two instruction mixes. (It should be noted that SCP performance has been stated to be 1.1 to 1.2 MIPS on the Gibson Mix, under unspecified conditions.).

National Series 32000 (NS32000)

The National NS32000 family is one of the leading families of 32-bit CPUs. All CPUs are based on a 32-bit (internal) bus architecture. Comparison is made against two versions of the first generation CPU (16-bit and 32-bit data bus) and the second generation CPU with enhanced internal micro-architecture (NS32332). Figure 9 compares execution rate for the EW instruction mix. All data for 10 Mhz. versions (-10) is measured, performance for the 15 Mhz. version (-15) is a projection. (Ref. 12)

Microprogrammed Bi-polar

The microprogrammed bi-polar bit-slice microprocessor has historically (10+ years) represented the standard for high performance processing. In addition to its basic high clock rate, its hardware architecture and instruction set can be customized to any particular application (i.e., algorithm). The track update algorithm used in this evaluation was originally implemented in a bi-polar bit-slice (2901 based) microprocessor. For this evaluation, the algorithm was coded in assembly language for execution on the AYK-14 and modified to use several new supplemental instructions for execution on the VHSM. Comparison is therefore being made on a functional equivalent algorithm rather than identical coding as in other previous comparisons.

Figure 10 shows comparative hardware execution times. The special purpose micro-coded bi-polar processor is approximately 2.5 times faster in execution than the VHSM. A portion of this difference can be attributed to the fact that the bit-slice architecture word length was customized to this specific application, while the VHSM required the use of (slower) double precision instructions for some of the signature parameters.

VHSIC Signal Processor

The first generally programmable Very High Speed Integrated Circuit (VHSIC) signal processor was developed by TRW as part of the VHSIC EW Brassboard. Its architecture incorporated six (6) VHSIC Phase 1 chip types designed and implemented using 1.25 micron feature size technology. The CPU contains three VHSIC arithmetic chips: two registered arithmetic logic units (RALUs) and a multiply-accumulate chip (MAC). The processor is programmable at both the macro and micro instruction level. The EW Brassboard is currently undergoing evaluation by NRL.

Figure 9: NS32000 Exec. Comparison

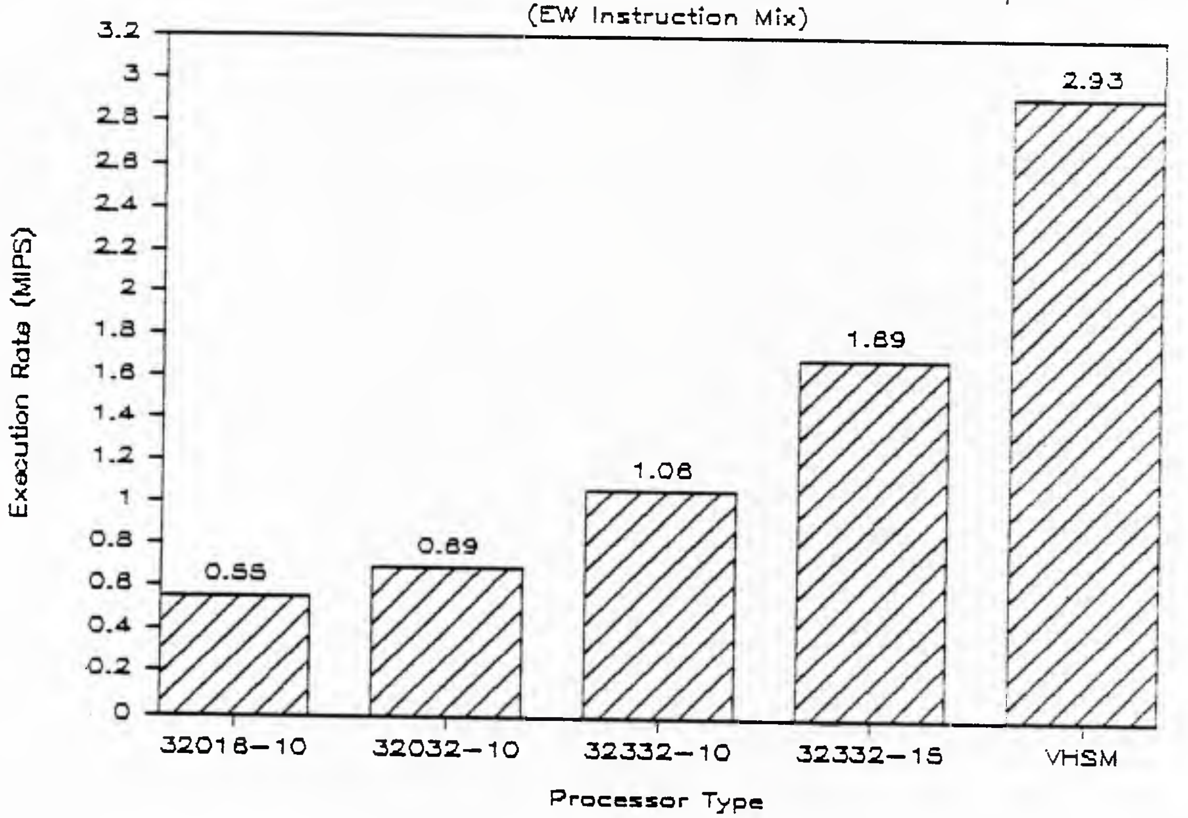
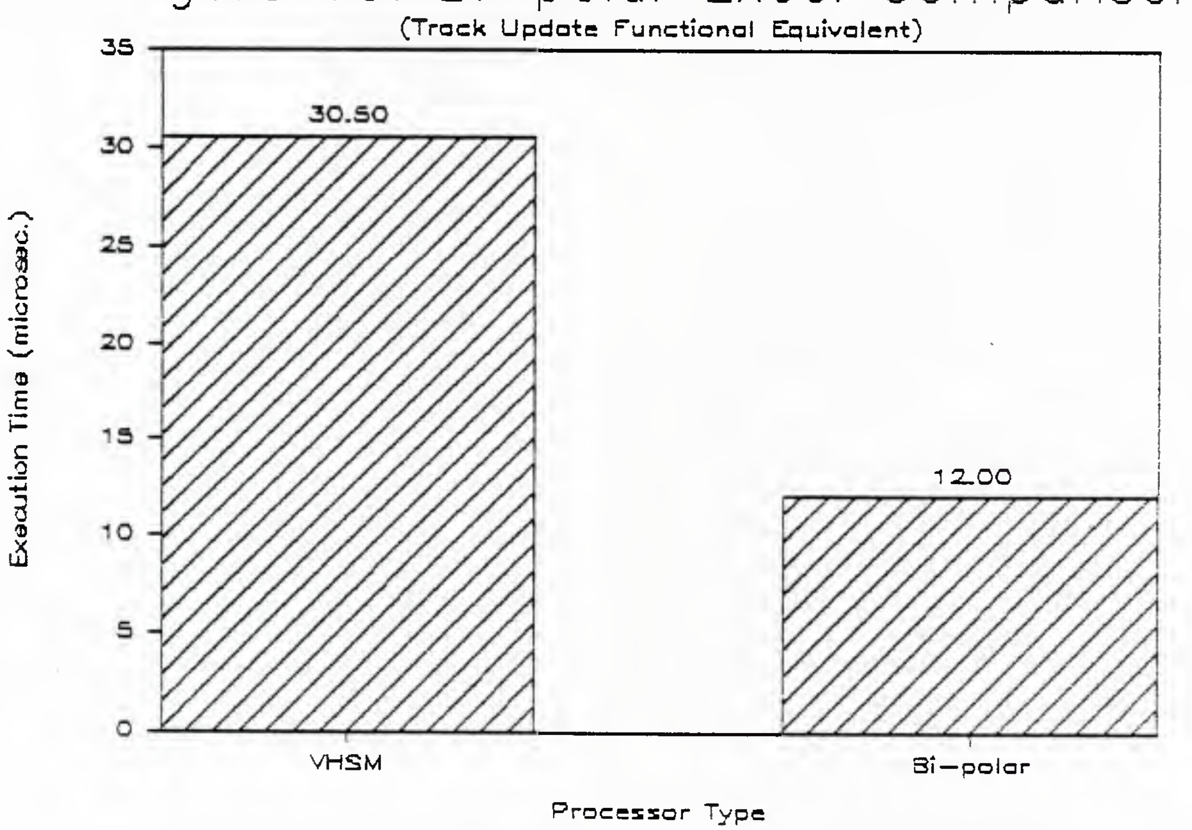


Figure 10: Bi-polar Exec. Comparison



Part of the Brassboard evaluation involved translating the Gibson mix and the EW instruction mix into VHSIC macro instructions. Some of the mix instructions do not translate, so similar or nearest equivalent instructions were used. Also, programming at the macro instruction level does not allow access to the dual RALU resources of the CPU, and therefore cannot demonstrate full potential of the architecture.

Actual hardware execution was measured by counting the number of clock cycles required to execute the test mixes. Figures 11 and 12 compare execution rates for the Gibson mix and the EW instruction mix, respectively. Performance of the VHSM is quite comparable to the VHSIC signal processor, with the VHSM slightly poorer in general purpose applications (the Gibson mix) and slightly better in EW applications (the EW instruction mix).

PROGRAMMABILITY AND LANGUAGE DEPENDENT VARIATIONS

One of the goals of the evaluation project was to demonstrate programmability of the VHSM breadboard. All test programs were written using MTASS, the Navy standard support software package. Programs were written in AYK-14 assembly language and the higher order languages CMS-2 and FORTRAN. One algorithm was written in two language formats to demonstrate software impact on execution performance and quantity and quality of resultant code. In addition, several programs were written incorporating several of the new instruction extensions unique to the VHSM. These routines are very small samples to demonstrate language dependent phenomena but not to quantify it. It should be recognized that programmer ability and experience can also significantly impact these same aspects of resultant code.

EW Tracker Algorithm

Implementation of the EW tracker algorithm in two different language formats permits comparison of language dependent impact on execution performance. Figure 13 shows execution performance expressed in terms of instruction execution rate. Such a comparison based on instruction execution rate is misleading because the different language formats result in different sizes of executable code. In fact, the CMS-2 compiled code was more "efficient" in terms of the resultant number of instructions. The assembly language version of the algorithm actually resulted in 11% more code being generated.

Figure 14 presents a more meaningful comparison based on the execution frequency of the algorithm. In both figures, data is presented for actual hardware execution in the VHSM and for simulated execution in the AYK-14 (MTASS SIM.). The assembly language version was measured to be 38% faster than the CMS-2 version. By contrast, the MTASS simulation projected that the assembly language version would only be 11% faster (in the AYK-14) than the CMS-2 version.

Fig. 11: VHSIC Macro Exec. Comparison

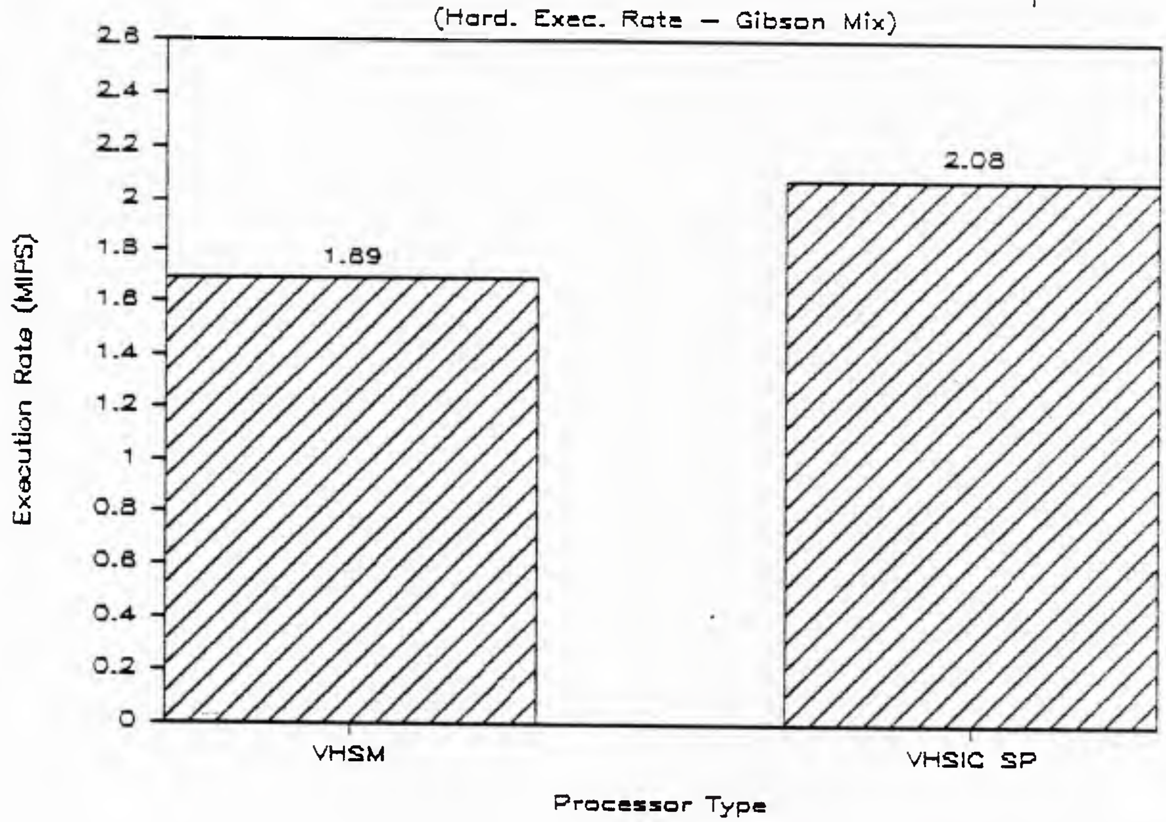


Fig. 12: VHSIC Macro Exec. Comparison

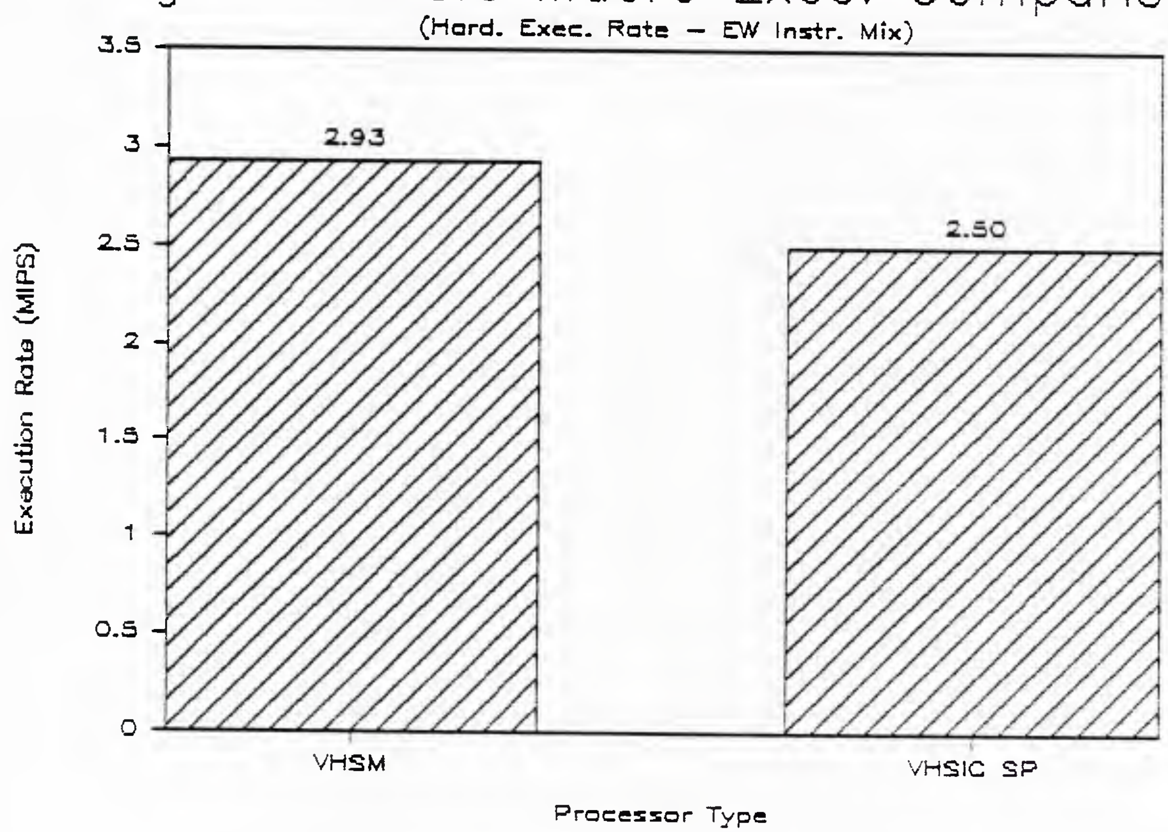


Figure 13: Lang. Dependent Exec. Rate

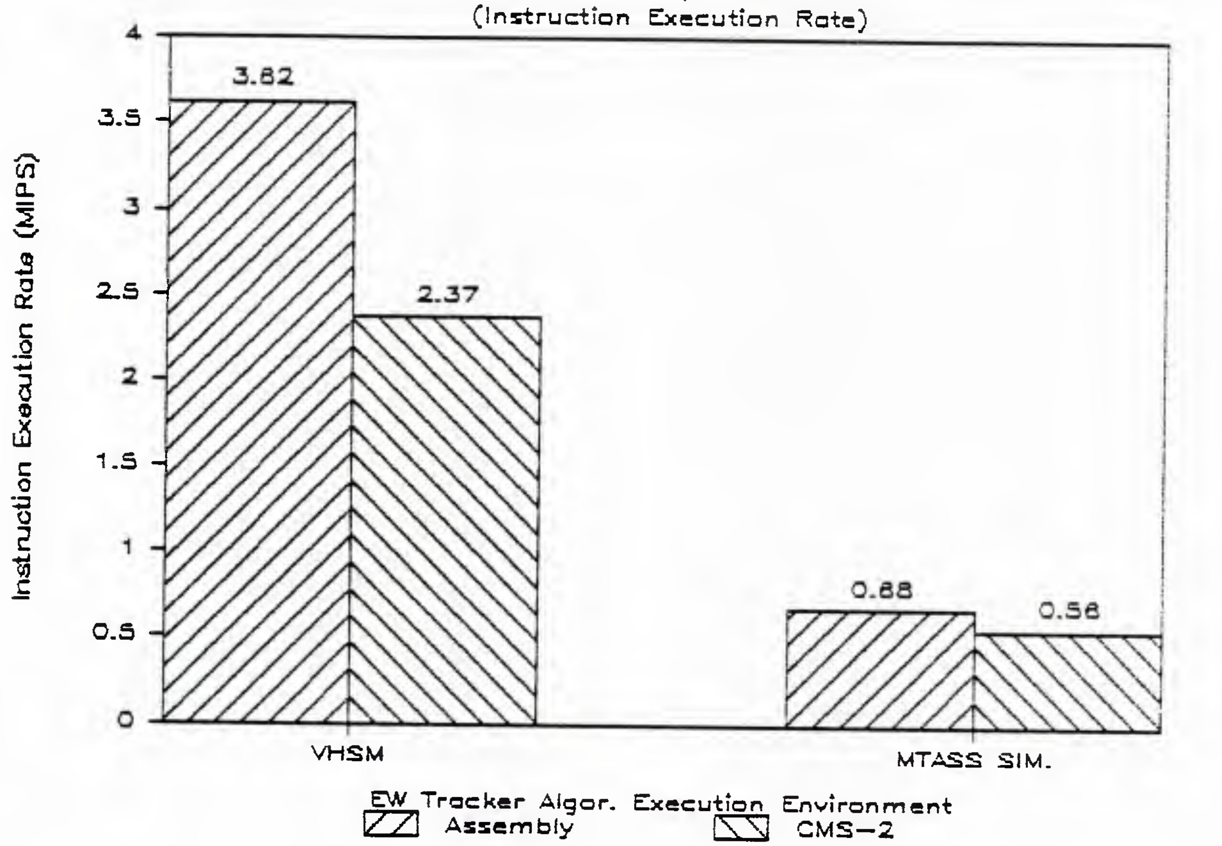
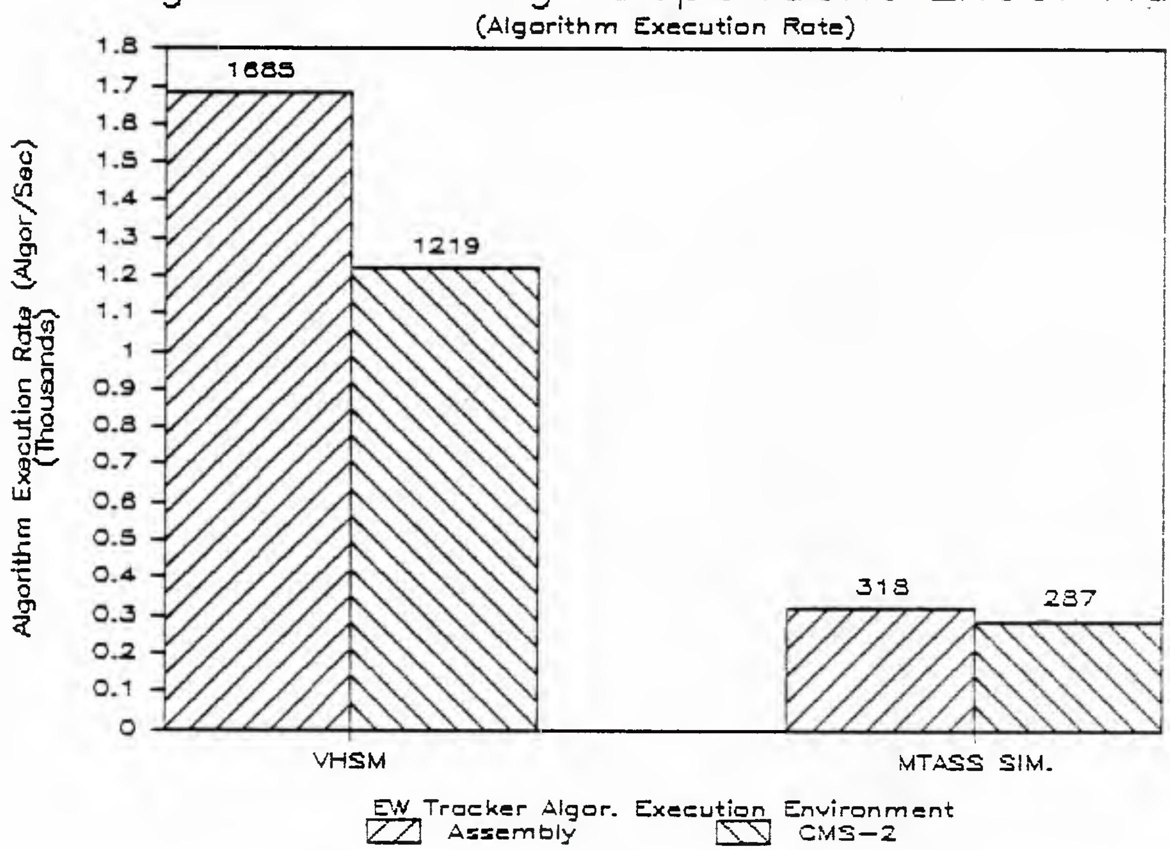


Figure 14: Lang. Dependent Exec. Rate



EW Sort Algorithm

The EW sort algorithm was selected to demonstrate the use of several "new" instructions which were implemented in the VHSM to extend the AYK-14 instruction set (PLUS), primarily for EW applications. The primary new instruction of interest in this application was one of six new compare-between-limits instructions. The one used here calls for the upper and lower comparison limits to be held in consecutive register locations. Use of this special purpose instruction resulted in coding which required the execution of approximately 1/3 fewer instructions in execution of the algorithm (299 versus 458). This reflects a desirable resultant feature of more compact and readable code.

Figure 15 shows execution performance expressed in terms of instruction execution rate. Because the execution time of this new instruction (0.540 usec.) is several times greater than the optimized primitive instructions (0.180 usec.), the resultant effective instruction execution rate is significantly lower than code written in normal AYK-14 assembly language. Figure 16 shows that when a more meaningful comparison is made based on the algorithm execution rate, the algorithm version employing the new instruction is found to be actually faster (approximately 2%) than the normal AYK-14 format version.

Track Update Algorithm

The track update algorithm was also used to demonstrate use of several "new" instructions. The new instructions of interest in this application were enhanced branch (jump) and direct memory instructions. These instructions can be expected to contribute to ease of programmability and some reduction in the size of executable code, but not to the extent demonstrated with the EW sort algorithm. More improvement could be achieved by restructuring the algorithm to take advantage of the (potentially) more powerful EW oriented compare-between-limits instructions. Only a single language format version (PLUS) was executed so no direct language dependent comparison can be made.

DATA DEPENDENT VARIATIONS

One application program was executed with different data sets as inputs. These data sets represent different data conditions which can impact performance for decision oriented algorithms, such as found in typical EW applications. This routine was again only intended to demonstrate the phenomena.

EW Tracker Algorithm

Four data tables were created for use with this algorithm to represent typical conditions encountered by a track algorithm: no data, small error, large error, and lost track. Figure 17 shows that the instruction execution rate shows minimal data dependence. Figure 18 presents a more meaningful comparison based on the execution time of the algorithm. The execution time can be seen to vary over a range of 2.2:1 for the assembly language version and over a range of 4.3:1 for CMS-2 version.

Figure 15: Enhanced Lang. Exec. Rate

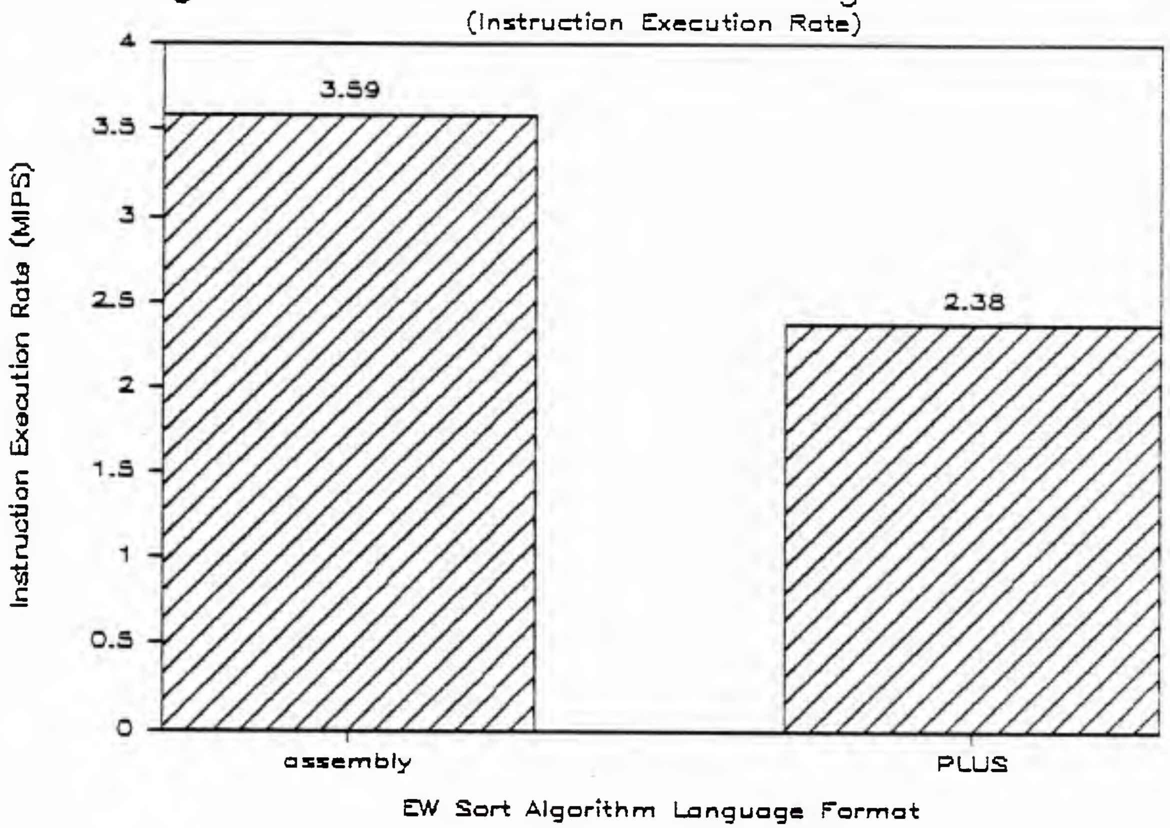


Figure 16: Enhanced Lang. Exec. Rate

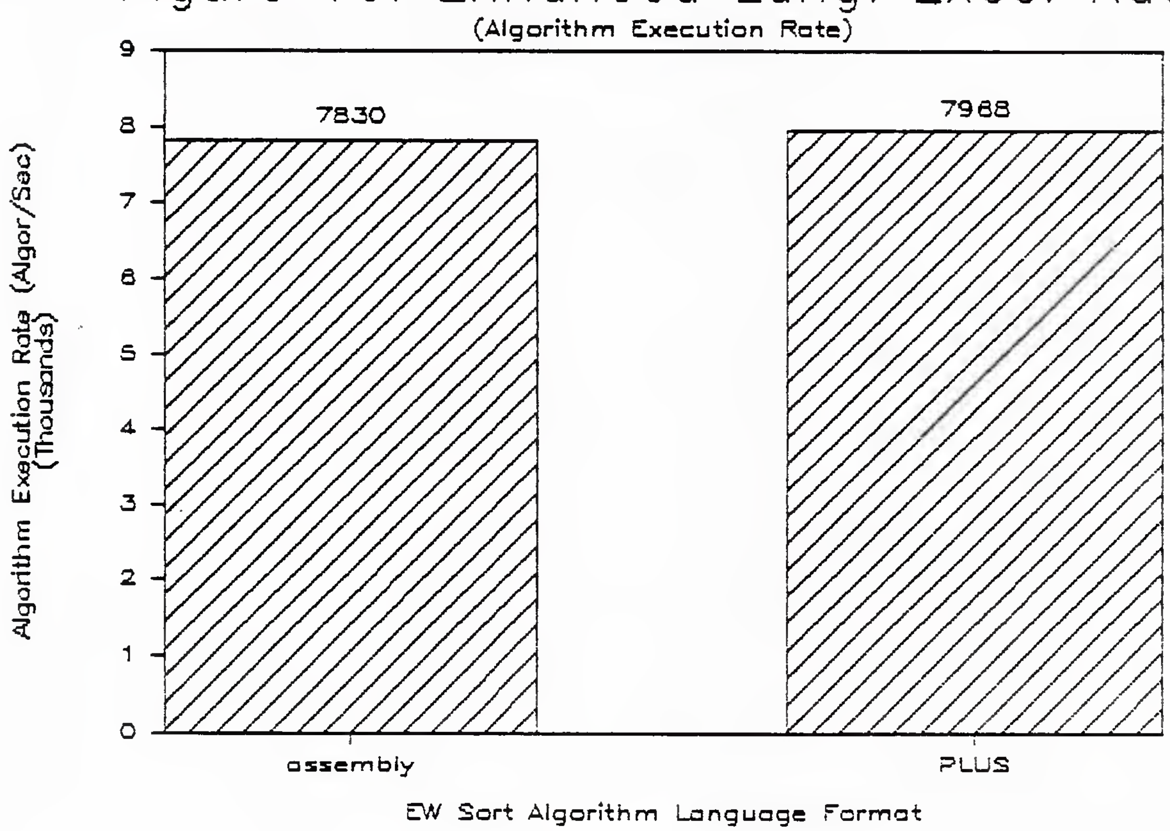


Figure 17: Data Dependent Exec. Rate

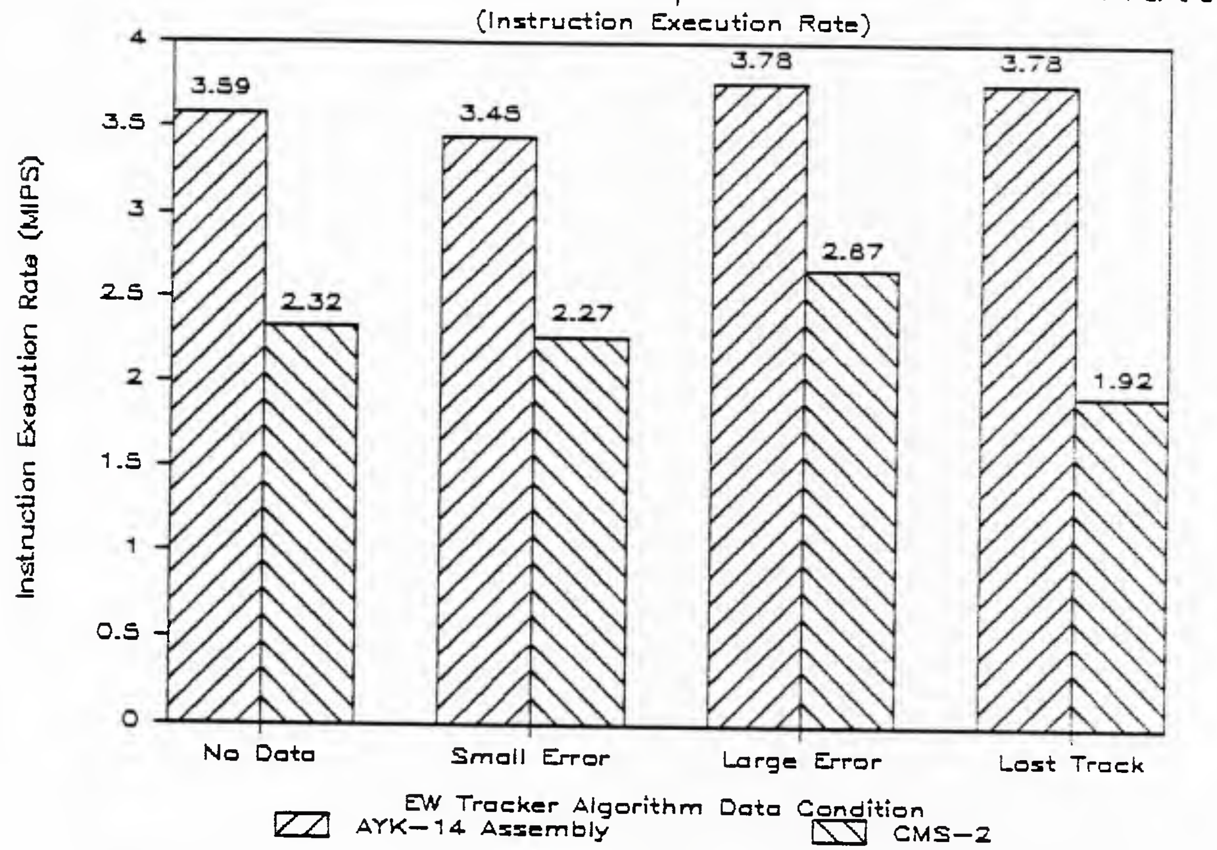
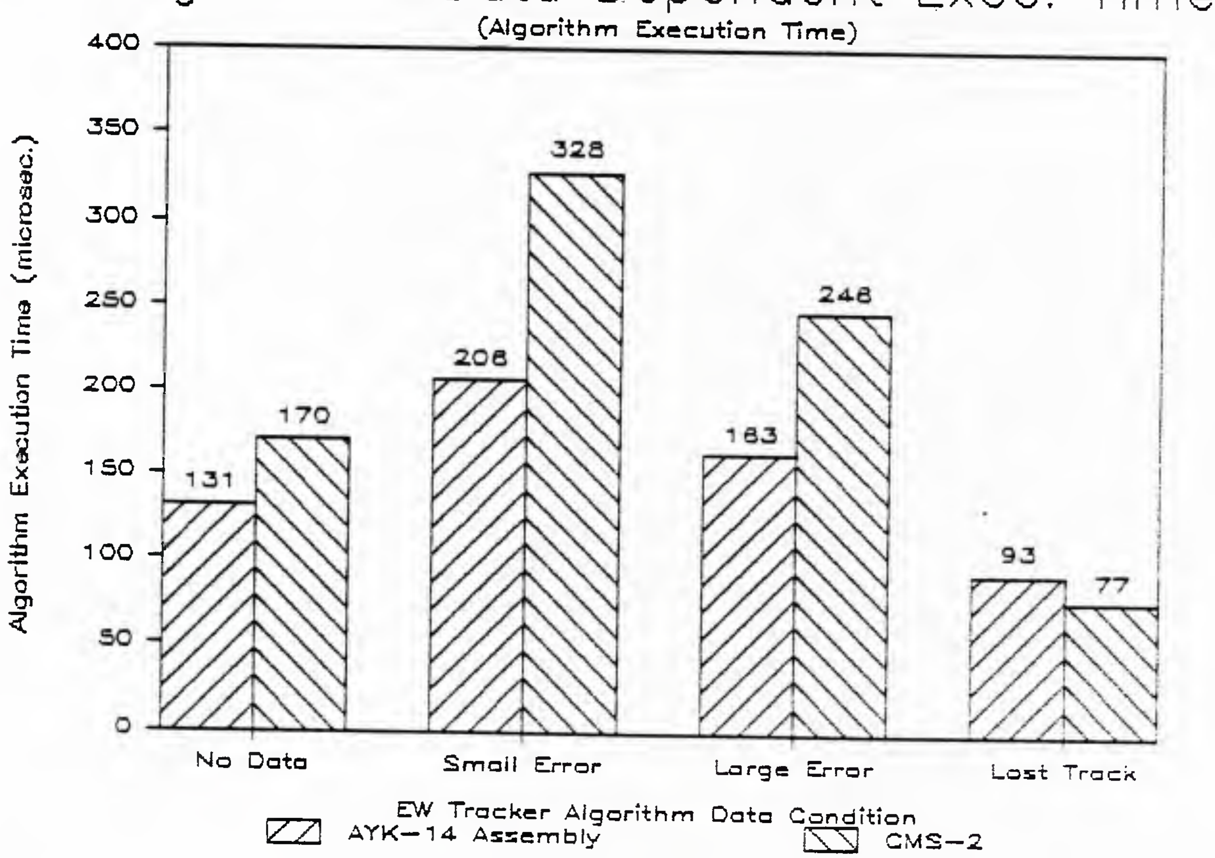


Figure 18: Data Dependent Exec. Time



SUMMARY AND CONCLUSIONS

The VHSM evaluation project undertook evaluation of the VHSM first generation breadboard for both its execution performance and its cost effectiveness. A series of test programs comprised of instruction mixes, benchmarks, and application programs were developed to carry out the project. Performance of the VHSM was demonstrated by actual hardware execution of test programs. Comparison was made by both hardware and simulated execution of an AYK-14 computer and other processors. Cost effectiveness was demonstrated by creating test programs based on familiar instruction sets and higher order languages, and implemented using Navy standard support software.

Instruction execution rate of the VHSM was demonstrated to be in the 2.9 to 3.6 MIPS range for EW applications and 1.7 MIPS for the more general purpose Gibson mix. Performance improvement achieved by using the split memory configuration was shown to be as much as 11% but quite application dependent. Direct comparison of the VHSM to the current AYK-14 computer showed that the VHSM is capable of achieving performance improvement ranging from 3.0 to 4.9.

Projected comparisons made against the new AYK-14 SCP show that the VHSM can expect to achieve performance improvement ranging from 1.7 to 2.5. It has also been shown that the VHSM offers significantly better performance than the new 32-bit architecture commercial processors in EW applications. However, microprogrammed bi-polar microprocessors can significantly outperform the VHSM when custom designed and optimized for execution of a specific algorithm. It was also shown that the VHSM can be competative (performance wise) with processors implemented using the new VHSIC implementation technology.

Test routines were written in different language formats to demonstrate ability to support familiar instruction set architectures and support software. Several routines were executed in different formats to demonstrate language dependent variations. It was demonstrated that different language formats will result in different sized executable code, but that the shortest code is not necessarily the fastest to execute. It was also shown that comparing the overall algorithm execution rate is more meaningful than comparing just the instruction execution rate.

Several test programs demonstrated that existing Navy MTASS could be used to write programs incorporating the new extended instructions. It was further demonstrated that use of these instructions could result in code that was significantly more compact and readable. An example of data dependent variations was presented to show that algorithm execution time can vary significantly in decision oriented algorithms typical of EW applications.

ACKNOWLEDGMENTS

The author acknowledges the efforts and contributions of those who made it possible to carry out this evaluation of the Very High Speed Microprocessor first generation breadboard:

Stephen J. Forde III of Sanders Associates for creating, coding and executing the EW Instruction Mix, the EW Tracker Algorithm, and the EW Sort Algorithm in the VHSM.

Roy Robinson of Quest Research Corp. for coding and executing the remaining test programs in the VHSM, and programs in the AYK-14 and VHSIC signal processor.

Sukumal Telepatra for compiling the projected performance for additional processors.

The NRL Electronic Warfare Support Measures (ESM) Branch management for guidance and direction throughout the program.

Mr. R. J. Orsino (formerly of the ESM Branch) for his guidance and direction throughout the program.

REFERENCES

1. R. M. Christiansen, "A Low Life-Cycle Cost EW Microprocessor," 1982 Government Microcircuit Applications Conference (GOMAC-82)

2. "1981 Technology Forecast - Software," Electronic Design, Jan. 8, 1981

3. R. M. Christiansen, "An EW Microprocessor for Technology Insertion," 1984 Government Microcircuit Applications Conference (GOMAC-84)

4. Boris Beizer, "Micro-Analysis of Computer System Performance," Van Norstrand Reinhold Co. New York, 1978

5. Edwin M. Drogin, "Pipeline Architecture Battles Array," Microwave Systems News, V. 10, No. 10, p. 92 1980

6. Gibson, J. C., "The Gibson Mix," IBM Tech. Rept. TR00.2043 (June 1970)

7. Ferrari, D., "Computer Systems Performance Evaluation," Prentice-Hall, Englewood Cliffs, NJ, 1978

8. Sanders Associates, "Design Microprocessor Architecture for EW Applications," contract No. N00014-81-C-2317 Final Report, Feb. 26, 1982

9. The NS16000 Benchmarks, National Semicondutor Corp., August 1983.

10. Sanders Associates, "High-Speed Microprocessor Breadboard Test Report," contract No. N00014-82-C-2347, 20 July, 1983

11. AN/AYK-14(V) Navy Standard Airborne Computer Technical Description, Control Data Corp. document G13693, Dec. 1984

12. National Semiconductor, private communication, June 30, 1986

DEPARTMENT OF THE NAVY

AVAL RESEARCH LABORATORY
Washington, D.C. 20375-5000

OFFICIAL BUSINESS
NALTY FOR PRIVATE USE, $300

POSTAGE AND FEES PAID
DEPARTMENT OF THE NAVY
DoD-316
THIRD CLASS MAIL

U.S.MAIL ®

Superintendent
U.S. Naval Postgraduate School
Monterey, CA 93940
Attn: Tech. Library